ESTIMATING CONFIDENCE INTERVALS ON ACCURACY IN CLASSIFICATION IN MACHINE LEARNING

By

Jesse Zhang, B.S. Computer Science, B.S. Mathematics

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of

Masters

in

Statistics

University of Alaska Fairbanks

April 2019

APPROVED:

Dr. Julie McIntyre, Committee Chair
Dr. Ronald Barry, Committee Member
Dr. Scott Goddard, Committee Member
Dr. Anthony Rickard, Chair
Department of Mathematics and Statistics
Dr. Leah Berman, Dean
University of Alaska Fairbanks
Dr. Michael Castellini, *Dean of the Graduate School*

Abstract

This paper explores various techniques to estimate a confidence interval on accuracy for machine learning algorithms. Confidence intervals on accuracy may be used to rank machine learning algorithms. We investigate bootstrapping, leave one out cross validation, and conformal prediction. These techniques are applied to the following machine learning algorithms: support vector machines, bagging AdaBoost, and random forests. Confidence intervals are produced on a total of nine datasets, three real and six simulated. We found in general not any technique was particular successful at always capturing the accuracy. However leave one out cross validation had the most consistency amongst all techniques for all datasets.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Special thanks to the arctic region supercomputing center for allowing me to run my simulations.

1 Introduction

Given a data set, one might utilize a myriad of machine learning algorithms to train and predict on it. Each machine learning algorithm has its own advantages and disadvantages, such as performing better or worse based on the distribution and type of data. Knowing which machine learning algorithm to use requires a good knowledge of the algorithm and an understanding of the dataset. The objective of this paper is to develop a simple way of ranking various machine learning algorithms against each other.

Machine learning algorithms are different from traditional statistical techniques because they don't place any assumptions on the random variables, nor do they offer much in the way of metrics for ranking how well these models fit, such as AIC or R^2 that are used to compare logistic or linear regression models.

Accuracy is a commonly used metric for ranking classification models. Here we will be focusing on accuracy in the scope of binary classification. Given a training and testing set we train the algorithm on the training set and use it to predict on the testing set. The total of number correct predictions over the total size of the testing set is called accuracy. A common way to rank machine learning algorithms is to compare their accuracies on the testing set. The question that arises is whether or not accuracy alone is a good ranking metric. We propose augmenting this metric by producing 95% confidence intervals on the accuracy of machine learning algorithms. In this context, a 95% CI suggests that if we were to obtain one hundred different training and testing sets from the same population, and developed 100 confidence intervals, approximately 95 of them would contain the true accuracy averaged over all possible training and testing sets selected from the dataset. CIs on accuracy can be used to rank algorithms, as if they do not overlap we can say the algorithms are significantly different in terms of accuracy.

For this project we select three common machine learning algorithms: Support Vector Machines, Bagging AdaBoost, and Random Forests. For each algorithm we use three differ-

ent techniques to estimate a confidence interval on accuracy: bootstrapping, leave one out cross validation, and conformal prediction. We compare techniques to see which can produce the best estimate. Bootstrapping is a commonly used statistical technique used for developing a confidence interval. Leave one out cross validation is a technique that involves training on all but one data point and predicting on it. It is used for machine learning algorithms and many other papers on this topic have used it for estimating accuracy(*Vanwinckelen and Blockeel* [2011] and *Kim* [2009]). Conformal prediction was developed by *Shafer and Vovk* [2007] and is a relatively new technique which we include to evaluate its ability to estimate accuracy.

This paper compares machine learning algorithms and confidence interval methods for a number of different data sets, both real and simulated. Real datasets are chosen such that they contain a single binary response, have a sample size of over 10,000 samples, and have no missing or incomplete entries. Simulated datasets are generated so we can compare techniques' ability to estimate confidence intervals on accuracy under different conditions.

The paper is organized as follows. In section 2 we provide background on the machine learning algorithms and the confidence interval estimation techniques that are the focus of this paper. Section 3 outlines the methodology of our study and the datasets used. Results are reported in section 4.

2 Techniques

2.1 Bootstrapping

Bootstrapping is a widely used technique and was introduced in 1979 in *Bootstrap Methods: Another Look at the Jackknife* by *Efron* [1979]. The general bootstrap method for computing a CI for a parameter θ works as follows. Given a random sample of size n, a large number B of bootstrap samples are generated by resampling from the original sample. Sampling is done with replacement, and the size of each bootstrap sample is n. An estimate of the parameter θ is computed for each bootstrap sample, yielding a sample of estimates, $\hat{\theta}_1, \hat{\theta}_2, \cdots \hat{\theta}_B$. Summaries on the sample of bootstrap estimates are used for inference about θ. For example, endpoints for a 95% CI are often identified as the 2.5 and 97.5 percentile values of the sample.

To obtain a confidence interval on accuracy we will be using a variation of the bootstrap developed by *Rinaldo et al.* [2016]. We first take the given dataset and split 70% into a training set and the remaining 30% into a testing set, which we will refer to as the "CI testing set". Next we generate a number of bootstrap samples from the training set. On each bootstrap sample we train each of the three machine learning algorithms, and use the results of the training to predict on the CI testing set. We then compute accuracy of the predictions made for each machine learning algorithm for each bootstrap sample. Finally we obtain the 2.5% percentile and 97.5% percentile of the accuracy to obtain a 95% confidence interval for each machine learning algorithm. This split then re-sampling method is displayed in figure 2.1.

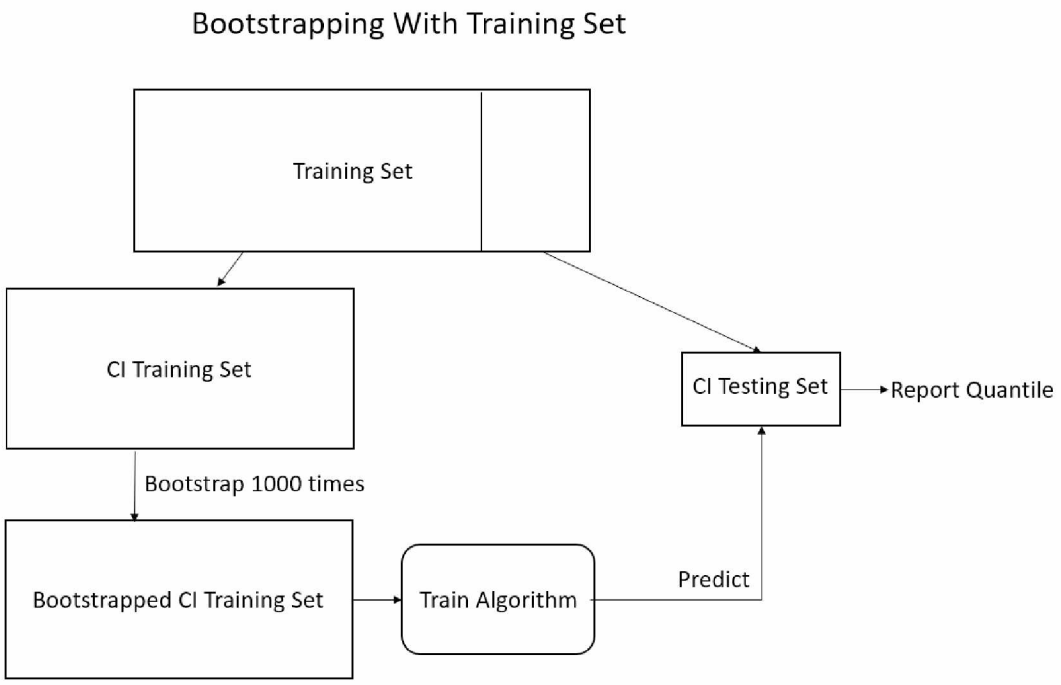

Figure 2.1: Bootstrapping

2.2 Leave one out cross validation

K-fold cross validation is a technique used in machine learning for model selection. First the dataset is shuffled randomly and partitioned into k groups. Then each unique group is removed from the dataset and designated as the testing set. We then train the model on the remaining $k-1$ groups and use the result to predict on the testing set. The accuracy is calculated and stored. This is repeated k times with each unique group designated as the testing set exactly once. This results in a prediction for each point in our dataset from which a CI is derived by using a normal approximation which is the form of $\hat{\theta} \pm 1.96 \times sd(\hat{\theta})$. There are several commonly used variations of K-fold cross validation. One is called leave one out cross validation, which sets k equal to the size of the dataset, meaning each unique group has only one sample. Another variation is n repeated K-fold cross validation which involves reshuffling the dataset and running K-folds again n times.

Kim [2009] and *Vanwinckelen and Blockeel* [2011] demonstrate that there is varying success when using K-fold cross validation to estimate error and produce a confidence interval on accuracy. *Vanwinckelen and Blockeel* [2011] shows that repeated K-fold cross validation, while decreasing the variance on estimated error, will not necessarily provide a more precise estimate of the accuracy. In K-folds we choose K small to attempt to minimize computational resources. In this case since we are more concerned on what K-fold's ability to estimate a confidence interval, we will choose to use the largest K possible which is leave one out. This is illustrated in figure 2.2. We choose K large because we want to have our predictions to be as accurate as possible to best estimate the true accuracy of the algorithm.

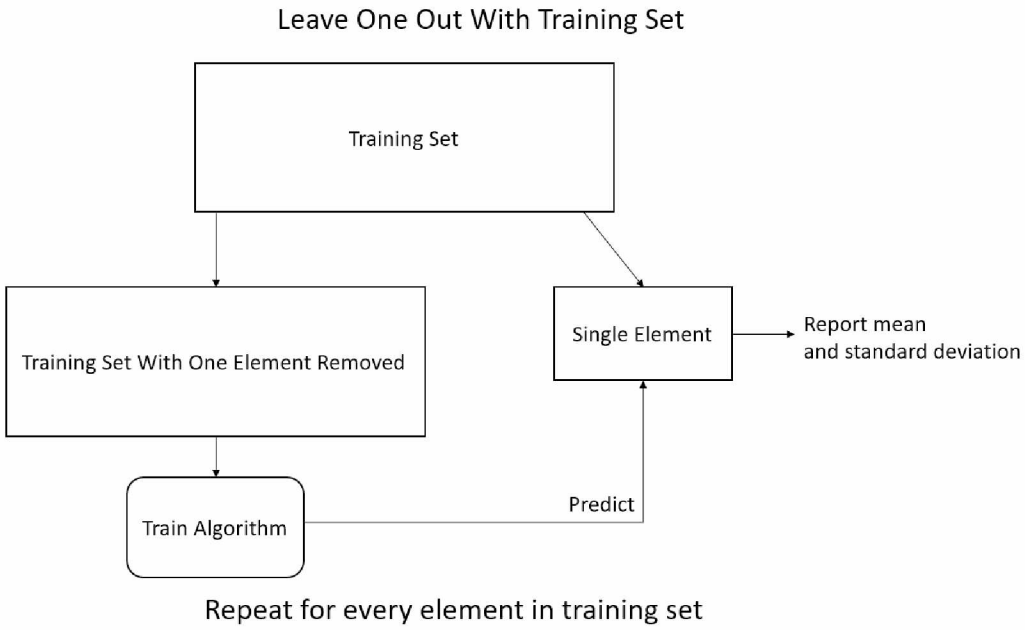

Figure 2.2: Leave One Out Cross Validation.

2.3 Conformal Prediction

Conformal prediction is a relatively new method first developed in 2007 by *Shafer and Vovk* [2007]. This method attaches a lower bound probability to a prediction made by a machine learning algorithm. This allows a measure of confidence to be attached to each prediction. The assumption placed on the dataset for conformal prediction to work is *ex-*

changeability. The random variables $X_1, X_2, \cdots X_n$ are said to be *exchangeable* if every possible sequence has the same joint distribution. For example X_1, X_2, X_3, X_4 must have the same joint distribution as X_4, X_3, X_1, X_2.

In this paper we use an implementation of conformal prediction called inductive conformal prediction(ICP). Before we begin describing how this procedure works we first define a nonconformity function. Denote a point in our data as $z = (\vec{x}, y)$ where $\vec{x}$ is the vector of all predictors and y is a factor response variable. Since this paper only focuses on binary classification y will take only two labels, for example either the value 0 or 1. Let B be a subset of our dataset containing the randomly selected points z_i. That is, if we were to randomly select n points for our subset B we would have $B = \{z_1, z_2, \cdots z_n\}$. Let us further define the function $gower(\vec{x_i}, \vec{x_j})$ which computes the Gower distance between $\vec{x_i}$ and $\vec{x_j}$. This is done via the package `gower` in R (*van der Loo* [2017]). Gower distance is a technique that allows for mixed data distance measuring (*Gower* [1971]). That is it measures the distance between two samples that consistent of factor, ordered, and continuous explanatory variables. Next, given the point $\tilde{z}$ and set B, we define the nonconformity function, A as,

$$A(B, \tilde{z}) = \frac{\min\{gower(\vec{x_i}, \vec{\tilde{x}}) : 1 \leq i \leq n; y_i = \tilde{y}\}}{\min\{gower(\vec{x_i}, \vec{\tilde{x}}) : 1 \leq i \leq n; y_i \neq \tilde{y}\}}.$$

Thus $A(B, \tilde{z})$ measures the ratio of the smallest Gower distance between $\tilde{x}$ at all points $z \in B$ with the same response value, to the smallest Gower distance between $\tilde{x}$ and all points with different response value. This nonconformity function is referred to the distance to nearest neighbors for classification (*Shafer and Vovk* [2007]).

Given a dataset of size n we split it into a training set of size m and a CI testing set of size $n - m$. The training set is then further split into a true training set of size $m - c$ and a calibration set of size c. Let us denote the values in our calibration set as $\{z_1, z_2, \cdots z_c\}$. Then for each z_i we compute the nonconformity function, $A(B, z_i)$. This produces a total of c nonconformity scores which we denote as $\{\beta_1, \beta_2, \cdots \beta_c\} = \hat{\beta}$. We then also train our

machine learning algorithm on our true training set, and predict on our CI testing set.

For each of the machine learning algorithm's predictions we wish to attach a measure of confidence. To describe this procedure we will focus on a single testing point $\hat{z}$. The predicted response for $\tilde{z}$ is either $y = 0$ or $y = 1$. Denote these possible outcomes as $\hat{z}_0$ and $\hat{z}_1$ respectively. We then calculate the nonconformity scores of $\hat{z}_0$ and $\hat{z}_1$ with our calibration set. Denote these scores as as $\hat{\beta}_0$ and $\hat{\beta}_1$. Now we convert these nonconformity scores into a desired confidence value. This is done by calculating the proportion of nonconformity scores that are greater than or equal to our $\beta \in \hat{\beta}$. Denote these proportions as $\hat{p}_0$ and $\hat{p}_1$. So given a prediction $\hat{y}$ we report the confidence of $(1 - \hat{p}_1)$ for $\hat{y} = 0$ and $(1 - \hat{p}_0)$ for $\hat{y} = 1$. This is repeated for every sample in the CI testing set. The overall method of conformal prediction is outlined in figure 2.3.

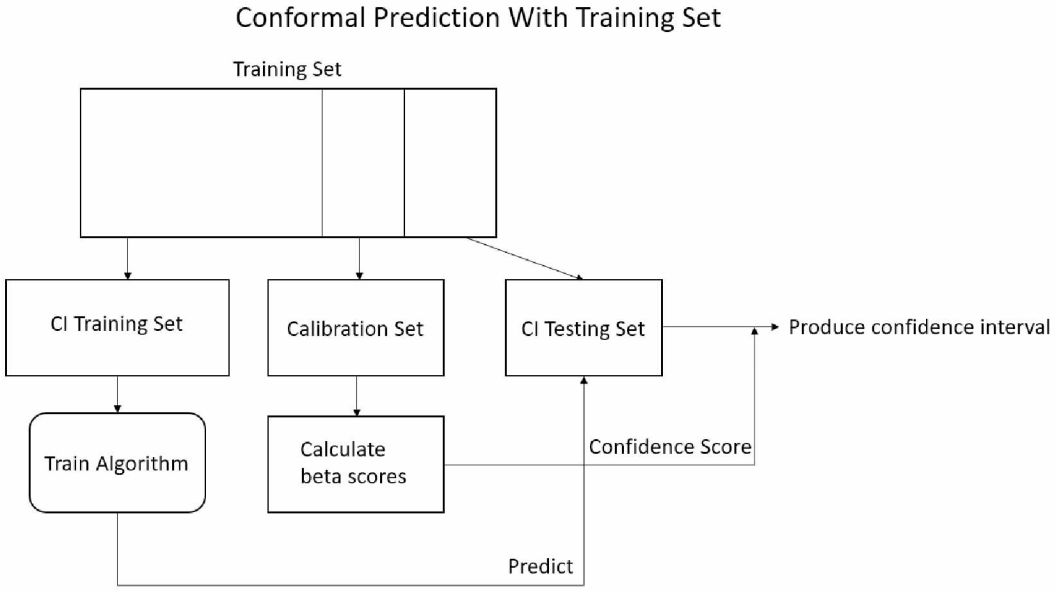

Figure 2.3: Splitting dataset into training set and testing set

Given our prediction and our associated confidence we wish to convert these values into a confidence intervals. While this method is not really designed for confidence intervals we propose the following extensions. In this experiment we try the following.

- Choosing $\alpha = 0.05$ and only accepting predictions where confidence is greater than or

equal to 0.95. Then compare the predicted value to the actual value and give a score of 1 if it is correct and 0 if it is not. If we do not accept the prediction a score of 0.5 is given.

- Choosing $\alpha = 0.1$ and only accepting predictions where confidence is greater than or equal to 0.9. Then compare the predicted value to the actual value and give a score of 1 if it is correct and 0 if it is not. If we do not accept the prediction a score of 0.5 is given.

- Compare the predicted value to the actual value. If the prediction is correct, take the maximum possible confidence. If the prediction is incorrect, take the minimum possible confidence.

- Compare the predicted value to the actual value. If the prediction is correct, report the confidence as its score.

We will then treat these results as a proportion and use a normal approximation to estimate the confidence interval. Each method has its advantages and disadvantages with the prevailing disadvantage being that none produce a true confidence interval. We proposed these extensions to see if we can estimate an interval that can capture the accuracy, and if it is successful we can use it for ranking.

2.4 Machine Learning Algorithms

The machine learning algorithms chosen to test in this experiment were taken from the *Top 10 algorithms in data mining* by *Wu et al.* [2008]. We have chosen to use Support Vector Machines, Bagged AdaBoost, and Random Forests. The idea is by choosing three algorithms in the top 10 algorithms, we can generalize our results to any classification machine learning technique in terms of comparing and ranking. Below is a quick summary and rundown of each chosen machine learning algorithm.

2.5 Support Vector Machines

Support vector machines are a supervised learning model that can be used for regression or classification analysis. The technique involves constructing a hyperplane that best separates the response variables. This hyperplane is then used for prediction on future points.

2.6 Bagged AdaBoost

AdaBoost is short for adaptive boosting and is another supervised learning model used for classification. Adaboost attempts to decrease the dimensionality of the problem by attempting to select only features that will improve the predictive power of the modeling. We are using Bagged Adaboost that allows us to prevent overfitting problems. In Bagging we randomly select our training set and sample set and train multiple models. Then when we are given a new point to predict on, it is given to each model. In the case of classification the most popular answer is given as the prediction.

2.7 Random Forest

A random forest attempts to fix the overfitting problem presented in training a single decision tree. In order to do this, multiple decision trees are constructed on different variations of the training dataset. Given a new point we wish to predict on, we put the point through all decision trees and the answer that given is done by popular voting.

3 Methodology

The following research was coded via *R*(*R Core Team* [2017]) and run in parallel on the UAF super computer using the package `snow` (*Tierney et al.* [2018]). No timing was run on the programs.

We select datasets with a large size where we define large as $n > 10,000$. We do this because we wish to minimize the effect of splitting the dataset multiple times. Additionally, since we have such a large size we will assume that this is the population. We then split the data initially into a training set and a testing set. The training set is sent to the various techniques we listed earlier to find the intervals. The machine learning algorithms are trained on the training set and then used to predict on the testing set, giving us the true machine learning accuracy on the population as seen in figure 3.1.

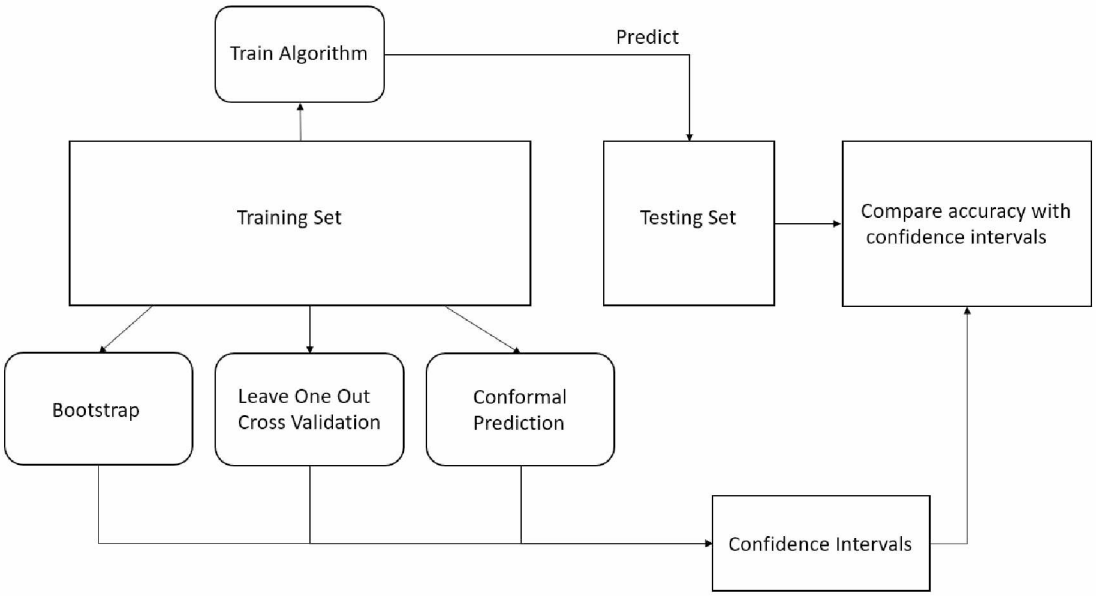

Figure 3.1: Training set gets sent to bootstrap, leave one out, and conformal prediction

3.1 Datasets

We use a total of 3 real life datasets and 6 simulated datasets in this experiment. This section will offer a quick review of the datasets. Table 3.1 contains a quick overview of the

number of variables and size of the datasets.

Dataset	Number of Predictors	Size	Simulated
Mammography	7	11183	No
Electricity	9	45312	No
EEG-eye State	15	14980	No
Small sample	8	140	Yes
Small sample with noise	13	140	Yes
Medium sample	8	3571	Yes
Medium sample with noise	13	3571	Yes
Large sample	8	17143	Yes
Large sample with noise	13	17143	Yes

Table 3.1: Dataset summary

3.2 Real Life Datasets

Real life datasets were obtained from OpenML or Open Machine Learning an online database of open license datasets (*Vanschoren et al.* [2013]). The three real life datasets we chose were Mammography, Electricity, and EEG-eye State.

3.2.1 Mammography

This dataset uses the creative commons license. The dataset consists of breast cancer screening and detections. This is an example of a dataset where the measured outcome is rare, including only 260 instances of cancer at a total of 11,183 observations. Since data are split three or four times depending on the technique, issues may arise with estimating accuracy due to having only 260 instanced of cancer. This dataset can be found on OpenML(*Vanschoren et al.* [2013]).

3.2.2 Electricity

The Electricity dataset was collected from the Australian New South Wales Electricity Market. The dataset identifies the fluctuations of energy transfers from New South Wales to Victoria, a neighboring state. The dataset is actually a time series as each point is taken

every 30 minutes with three of the variables indicating time (date,day,period). This dataset is under the creative commons license.

3.2.3 EEG-eye State

The EEG-eye state dataset is under the creative commons license it involves one continuous EEG measurement with the Emotiv EEG Neuroheadset. The predictor is whether the eye is closed or not, and the data is sampled over 117 seconds. A total of 14 EEG measurements are taken and used as explanatory variables. This dataset was specifically taken from the University of California, Irvine dataset repository (*Dua and Graff* [2017]).

3.3 Simulated Datasets

We simulate 6 datasets using the R function `twoClassSim` in the `caret` package (*Max Kuhn et al.*[2018]). The main idea is to compare performance of techniques on datasets of various sample size, and including various amounts of noise. We use 3 sample size levels, 140, 3571, 17143. Noise is defined as the inclusion of extraneous predictors, and we use 2 noise levels, 0 and 5 extraneous predictors. A single dataset is simulated for each sample size and noise level, resulting in 6 simulated datasets.

4 Results

4.1 Results

The results section is split up into 9 different tables, one for each corresponding dataset. Confidence intervals for each dataset are shown in Tables 4.1-4.8. 95% CI's are produced for each technique and algorithm. In addition, the success of each confidence interval in capturing the true accuracy is indicated in the column header, Success.

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.97956,0.98296)	0.98390	
	AdaBag	(0.97615,0.97998)	0.97973	✓
	Random Forest	(0.98382,0.98765)	0.98748	✓
Leave One Out	SVM	(0.97918,0.98505)	0.98390	✓
	AdaBag	(0.97574,0.98211)	0.97973	✓
	Random Forest	(0.9846,0.9896)	0.98748	✓
Conformal Prediction Score with default confidence	SVM	(0.97443,0.98573)	0.98390	✓
	AdaBag	(0.97382,0.98527)	0.97973	✓
	Random Forest	(0.98235,0.99154)	0.98748	✓
Conformal Prediction with $\alpha = 0.05$	SVM	(0.97626,0.98711)	0.98390	✓
	AdaBag	(0.97553,0.98656)	0.97973	✓
	Random Forest	(0.98293,0.99194)	0.98748	✓
Conformal Prediction with $\alpha = 0.1$	SVM	(0.97578,0.98674)	0.98390	✓
	AdaBag	(0.97480,0.98601)	0.97973	✓
	Random Forest	(0.98368,0.99247)	0.98748	✓
Conformal Prediction using max confidence	SVM	(0.99326,0.99846)	0.98390	
	AdaBag	(0.99402,0.99884)	0.97973	
	Random Forest	(0.99479,0.99921)	0.98748	

Table 4.1: Mammography dataset result

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.75205,0.75583)	0.75539	✓
	AdaBag	(0.75131,0.75552)	0.75414	✓
	Random Forest	(0.85968,0.86736)	0.89193	✓
Leave One Out	SVM	(0.74951,0.75899)	0.75539	✓
	AdaBag	(0.75405,0.76347)	0.75414	✓
	Random Forest	(0.88618,0.89307)	0.89193	✓
Conformal Prediction Score with default confidence	SVM	(0.71397,0.73195)	0.75539	
	AdaBag	(0.71646,0.73439)	0.75414	
	Random Forest	(0.76078,0.77771)	0.89193	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.61641,0.63585)	0.75539	
	AdaBag	(0.61731,0.63674)	0.75414	
	Random Forest	(0.63543,0.65466)	0.89193	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.66829,0.68708)	0.75539	
	AdaBag	(0.66983,0.68859)	0.75414	
	Random Forest	(0.7028,0.721)	0.89193	
Conformal Prediction using max confidence	SVM	(0.79072,0.80683)	0.75539	
	AdaBag	(0.79369,0.80971)	0.75414	
	Random Forest	(0.85116,0.86518)	0.89193	

Table 4.2: Electricity dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.63953,0.66974)	0.61273	
	AdaBag	(0.58741,0.63477)	0.60071	✓
	Random Forest	(0.89320,0.90751)	0.92989	
Leave One Out	SVM	(0.60966,0.62825)	0.61273	✓
	AdaBag	(0.59893,0.61762)	0.60071	✓
	Random Forest	(0.92365,0.93351)	0.92989	✓
Conformal Prediction Score with default confidence	SVM	(0.77507,0.80357)	0.61273	
	AdaBag	(0.74885,0.77854)	0.60071	
	Random Forest	(0.89238,0.91309)	0.92989	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.74558,0.77540)	0.61273	
	AdaBag	(0.72174,0.75251)	0.60071	
	Random Forest	(0.86220,0.88541)	0.92989	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.76101,0.79017)	0.61273	
	AdaBag	(0.73276,0.76311)	0.60071	
	Random Forest	(0.88955,0.91051)	0.92989	
Conformal Prediction using max confidence	SVM	(0.80930,0.836)	0.61273	
	AdaBag	(0.78090,0.80911)	0.60071	
	Random Forest	(0.94344,0.95853)	0.92989	

Table 4.3: EEG Eye dataset results

We can see in our real datasets that if our algorithm can achieve a high accuracy, as seen in the mammography dataset in table 4.1, then most techniques will be able to generate a proper interval. A problem arises when the accuracy is not high, as seen in table 4.2 and 4.3. Since the accuracy is low we see that only leave one out cross validation is able to estimate the true accuracy consistently. Note that most conformal prediction techniques in general produce confidence intervals that always overestimate or underestimate the accuracy for each dataset. It may be reliant on the way our data is distributed.

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.60714,0.89286)	0.75610	✓
	AdaBag	(0.42857,0.82143)	0.70732	✓
	Random Forest	(0.53571,0.82143)	0.87805	
Leave One Out	SVM	(0.80216,0.93522)	0.75610	
	AdaBag	(0.67316,0.84199)	0.70732	✓
	Random Forest	(0.75399,0.90257)	0.87805	✓
Conformal Prediction Score with default confidence	SVM	(0.44568,0.80432)	0.75610	✓
	AdaBag	(0.40706,0.77151)	0.70732	✓
	Random Forest	(0.42449,0.78655)	0.87805	✓
Conformal Prediction with $\alpha = 0.05$	SVM	(0.38813,0.75473)	0.75610	
	AdaBag	(0.38813,0.75473)	0.70732	✓
	Random Forest	(0.40706,0.77151)	0.87805	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.40706,0.77151)	0.75610	✓
	AdaBag	(0.38813,0.75473)	0.70732	✓
	Random Forest	(0.42624,0.78804)	0.87805	
Conformal Prediction using max confidence	SVM	(0.56231,0.89224)	0.75610	✓
	AdaBag	(0.50558,0.85156)	0.70732	✓
	Random Forest	(0.54314,0.87894)	0.87805	✓

Table 4.4: Simulated Small sample dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.64286,0.89286)	0.65854	✓
	AdaBag	(0.5,0.78571)	0.70732	✓
	Random Forest	(0.64286,0.96429)	0.85366	✓
Leave One Out	SVM	(0.76587,0.83838)	0.65854	
	AdaBag	(0.63954,0.81500)	0.70732	✓
	Random Forest	(0.81449,0.94308)	0.85366	✓
Conformal Prediction Score with default confidence	SVM	(0.40014,0.76544)	0.65854	✓
	AdaBag	(0.38642,0.75319)	0.70732	✓
	Random Forest	(0.38471,0.75165)	0.85366	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.35098,0.72044)	0.65854	✓
	AdaBag	(0.33277,0.70294)	0.70732	
	Random Forest	(0.35098,0.72044)	0.85366	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.35098,0.72044)	0.65854	✓
	AdaBag	(0.33277,0.70294)	0.70732	
	Random Forest	(0.35098,0.72044)	0.85366	
Conformal Prediction using max confidence	SVM	(0.51115,0.85573)	0.65854	✓
	AdaBag	(0.44924,0.80725)	0.70732	✓
	Random Forest	(0.49451,0.84316)	0.85366	

Table 4.5: Small sample with noise dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.80240,0.82510)	0.79085	
	AdaBag	(0.64082,0.66222)	0.66106	✓
	Random Forest	(0.86248,0.88919)	0.86928	✓
Leave One Out	SVM	(0.79462,0.82538)	0.79085	
	AdaBag	(0.65521,0.69198)	0.66106	✓
	Random Forest	(0.87522,0.89998)	0.86928	
Conformal Prediction Score with default confidence	SVM	(0.67576,0.74087)	0.79085	
	AdaBag	(0.63381,0.70129)	0.66106	✓
	Random Forest	(0.69458,0.75842)	0.86928	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.56437,0.63456)	0.79085	
	AdaBag	(0.55828,0.62864)	0.66106	
	Random Forest	(0.57251,0.64245)	0.86928	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.614,0.68240)	0.79085	
	AdaBag	(0.59152,0.66082)	0.66106	
	Random Forest	(0.62834,0.69609)	0.86928	
Conformal Prediction using max confidence	SVM	(0.76433,0.82233)	0.79085	✓
	AdaBag	(0.70912,0.77190)	0.66106	
	Random Forest	(0.79542,0.85012)	0.86928	

Table 4.6: Simulated Medium sample dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.77570,0.79706)	0.77965	✓
	AdaBag	(0.66222,0.68625)	0.68627	✓
	Random Forest	(0.86782,0.89586)	0.88235	✓
Leave One Out	SVM	(0.79338,0.82422)	0.77965	
	AdaBag	(0.65076,0.68764)	0.68627	✓
	Random Forest	(0.88109,0.90531)	0.88235	✓
Conformal Prediction Score with default confidence	SVM	(0.67360,0.73885)	0.77965	
	AdaBag	(0.63638,0.70373)	0.68627	✓
	Random Forest	(0.69388,0.75777)	0.88235	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.57251,0.64245)	0.77965	
	AdaBag	(0.54814,0.61875)	0.68627	
	Random Forest	(0.57658,0.64639)	0.88235	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.61605,0.68435)	0.77965	
	AdaBag	(0.58336,0.65295)	0.68627	
	Random Forest	(0.62629,0.69413)	0.88235	
Conformal Prediction using max confidence	SVM	(0.76356,0.82164)	0.77965	✓
	AdaBag	(0.69865,0.76221)	0.68627	
	Random Forest	(0.78726,0.84287)	0.88235	

Table 4.7: Simulated Medium sample with noise dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.81,0.81667)	0.81447	✓
	AdaBag	(0.655,0.67861)	0.67192	✓
	Random Forest	(0.8925,0.90195)	0.90373	
Leave One Out	SVM	(0.79599,0.81021)	0.81447	
	AdaBag	(0.66547,0.68225)	0.67192	✓
	Random Forest	(0.89771,0.90830)	0.90373	✓
Conformal Prediction Score with default confidence	SVM	(0.73805,0.76626)	0.81447	
	AdaBag	(0.68974,0.71955)	0.67192	
	Random Forest	(0.77281,0.7996)	0.90373	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.62152,0.65293)	0.81447	
	AdaBag	(0.60078,0.63255)	0.67192	
	Random Forest	(0.63863,0.66970)	0.90373	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.68869,0.71853)	0.81447	
	AdaBag	(0.65197,0.68275)	0.67192	✓
	Random Forest	(0.7179,0.74682)	0.90373	
Conformal Prediction using max confidence	SVM	(0.81333,0.83812)	0.81447	✓
	AdaBag	(0.75151,0.7792)	0.67192	
	Random Forest	(0.86014,0.88203)	0.90373	

Table 4.8: Simulated Large sample dataset results

Technique	Algorithm	95% Confidence Interval	True Accuracy	Success
Bootstrap	SVM	(0.79105,0.79717)	0.80475	
	AdaBag	(0.66073,0.68130)	0.67231	✓
	Random Forest	(0.88830,0.89831)	0.90335	
Leave One Out	SVM	(0.79075,0.80512)	0.80475	✓
	AdaBag	(0.65935,0.67620)	0.67231	✓
	Random Forest	(0.89976,0.91025)	0.90335	✓
Conformal Prediction Score with default confidence	SVM	(0.6755,0.7057)	0.80475	
	AdaBag	(0.63884,0.66992)	0.67231	✓
	Random Forest	(0.69861,0.72815)	0.90335	
Conformal Prediction with $\alpha = 0.05$	SVM	(0.56739,0.59960)	0.80475	
	AdaBag	(0.55622,0.58854)	0.67231	
	Random Forest	(0.58178,0.61383)	0.90335	
Conformal Prediction with $\alpha = 0.1$	SVM	(0.61440,0.64595)	0.80475	
	AdaBag	(0.59228,0.62417)	0.67231	
	Random Forest	(0.63446,0.66562)	0.90335	
Conformal Prediction using max confidence	SVM	(0.75877,0.78617)	0.80475	
	AdaBag	(0.69858,0.72813)	0.67231	
	Random Forest	(0.80934,0.83434)	0.90335	

Table 4.9: Simulated Large sample with noise dataset results

We see through the nine datasets that leave one out performed the best in capturing the true accuracy, though it did not succeed consistently, with sometimes the interval being underneath or above the true accuracy. Interestingly, leave one out cross validation intervals successfully captured the accuracy on all real datasets.

In general conformal prediction gave poor intervals, however this may be due to the fact that we have failed to transform a confidence score into an interval and further research on this may be required. The problem in the current approach is that if the data is messy we cannot produce a high confidence value and therefore any prediction, correct or wrong, will be scored badly. This weighs heavily on the interval and prevents an accurate interval.

Bootstrapping performed well on most experiments but also failed to capture the accuracy some of the time. This shows that while it may be able to estimate error it is not consistent enough and there may need to be a method that utilizes multiple data splits.

In the simulated data we see that in general the smaller sample size gave a larger interval, which may accurately capture the accuracy, but may necessarily not be useful in ranking. A

medium sample size compared with a large sample size shows that the intervals are around the same size, meaning that perhaps the requirement for $n > 10,000$ may not be that necessary. It seems that inputting noise levels into the simulated dataset did not effect the results.

It is important to note that while in general leave one out performed the best it is also the most computationally expensive. Let n be the size of the dataset, then for leave one out we have to train n times', this can lead to exceedingly long times on a computer. Bootstrapping limits itself to 1000 trainings in this experiment, though it may be increased for perhaps further accuracy. Finally, conformal prediction is the cheapest in terms of computation time to calculate, however, since the intervals often did not include the true accuracy, they cannot really be considered.

5 Conclusion

In conclusion it seems that leave one out or bootstrapping would be the best way to make intervals and use them for ranking. Further replications of this experiment should be done to test the validity of these methods to produce accurate enough intervals for ranking. The methods we proposed to extend conformal prediction have not successfully produced accurate intervals. However we believe there may be a way to successfully encode the confidence and transform it into an interval similar to *Wu et al.* [2008] paper on regression.

5.1 Future work

As stated, replication on the same datasets will be useful in determining whether or not these results are replicable with different data splits. Testing on additional mixed datasets will also be important as this experiment mainly focused on continuous predictors. Additionally, we should look further into encoding a confidence and transforming it into an interval, perhaps with ways that may estimate the separability of the data. Finally testing this experiment with more simulated data may be helpful as simulation was reliant heavily on one package, the `twoClassSim` in the package `Caret` in R (*Max Kuhn et al.*[2018]). Using one functions means that all our simulated datasets are generated through one type of algorithm.

References

Dua, D., and C. Graff (2017), UCI machine learning repository.

Efron, B. (1979), Bootstrap methods: Another look at the jackknife, *The Annals of Statistics*, *7*(1), 1–26, doi:10.1214/aos/1176344552.

Gower, J. C. (1971), A general coefficient of similarity and some of its properties, *Biometrics*, *27*(4), 857, doi:10.2307/2528823.

Kim, J.-H. (2009), Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap, *Computational Statistics & Data Analysis*, *53*(11), 3735–3745, doi:10.1016/j.csda.2009.04.009.

Max Kuhn, C. f. J. W., S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. (2018), *caret: Classification and Regression Training*, r package version 6.0-79.

R Core Team (2017), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Rinaldo, A., L. Wasserman, M. G'Sell, and J. Lei (2016), Bootstrapping and sample splitting for high-dimensional, assumption-free inference.

Shafer, G., and V. Vovk (2007), A tutorial on conformal prediction.

Tierney, L., A. J. Rossini, N. Li, and H. Sevcikova (2018), *snow: Simple Network of Workstations*, r package version 0.4-3.

van der Loo, M. (2017), *gower: Gower's Distance*, r package version 0.1.2.

Vanschoren, J., J. N. van Rijn, B. Bischl, and L. Torgo (2013), Openml: Networked science in machine learning, *SIGKDD Explorations*, *15*(2), 49–60, doi:10.1145/2641190.2641198.

Vanwinckelen, G., and H. Blockeel (2011), On estimating model accuracy with repeated cross-validation., *On Estimating Model Accuracy with Repeated Cross-Validation.*

Wu, X., V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg (2008), Top 10 algorithms in data mining, *Knowledge and Information Systems*, *14*(1), 1–37, doi:10.1007/s10115-007-0114-2.